(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 108170503 B
(45)授权公告日 2022.04.12

(21)申请号 201810069208.7
(22)申请日 2018.01.24
(65)同一申请的已公布的文献号
申请公布号 CN 108170503 A
(43)申请公布日 2018.06.15
(73)专利权人 腾讯科技(深圳)有限公司
地址 518000 广东省深圳市南山区高新区科技中一路腾讯大厦35层
(72)发明人 黄源超 龙海 何家明 赖祖泽 周锐 吴智文 杜国阳 曹琛 谢宗祥
(74)专利代理机构 北京派特恩知识产权代理有限公司 11270
代理人 王姗姗 张颖玲
(51)Int.Cl.
*G06F 9/451*(2018.01)
*G06F 9/445*(2018.01)
(56)对比文件
CN 106445630 A,2017.02.22
CN 103488466 A,2014.01.01
CN 106293880 A,2017.01.04
CN 106951294 A,2017.07.14
CN 102123196 A,2011.07.13
CN 103327117 A,2013.09.25
CN 106970828 A,2017.07.21
CN 103488466 A,2014.01.01
US 2015186625 A1,2015.07.02
WO 2016176059 A1,2016.11.03
逝去的湮火.windows10相关资料(四).《https://mp.weixin.qq.com/s/0RnIXT3KBVfP4QXzdFpaBg》.2015,
审查员 曹永敏

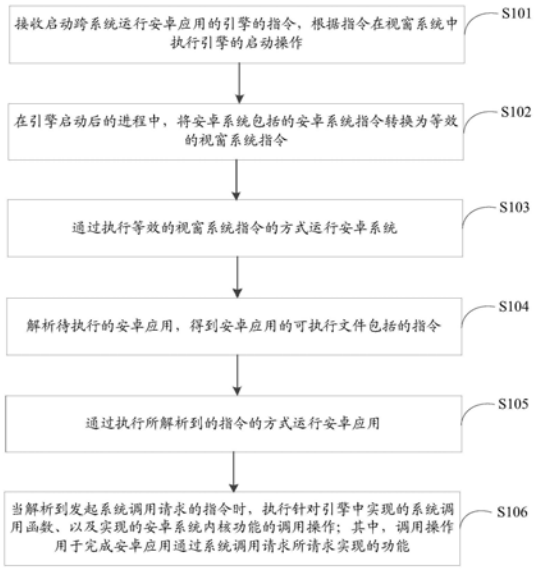
权利要求书3页 说明书17页 附图8页

(54)发明名称

一种跨系统运行安卓应用的方法、终端及存储介质

(57)摘要

本发明实施例公开了一种跨系统运行安卓应用的方法、终端及存储介质，该方法包括：接收启动跨系统运行安卓应用的引擎的指令，根据指令在视窗系统中执行引擎的启动操作；在引擎启动后的进程中，将安卓系统包括的安卓系统指令转换为等效的视窗系统指令；通过执行等效的视窗系统指令的方式运行安卓系统；解析待执行的安卓应用，得到安卓应用的可执行文件包括的指令；通过执行所解析到的指令的方式运行安卓应用；当解析到发起系统调用请求的指令时，执行针对引擎中实现的系统调用函数、以及实现的安卓系统内核功能的调用操作；其中，调用操作用于完成安卓应用通过系统调用请求所请求实现的功能。




CN 108170503 B

1.一种跨系统运行安卓应用的方法,其特征在于,包括:

接收启动跨系统运行安卓应用的引擎的指令,根据所述指令在视窗系统中执行所述引擎的启动操作;在所述引擎启动后的进程中,将所述安卓系统包括的安卓系统指令转换为等效的视窗系统指令;

通过执行所述等效的视窗系统指令的方式运行所述安卓系统;解析待执行的安卓应用,得到所述安卓应用的可执行文件包括的指令;

通过执行所解析到的指令的方式运行所述安卓应用;

当通过所述视窗系统内核,检测到所述安卓应用发起系统调用请求时,接收所述视窗系统的内核发送的所述系统调用请求;

根据所述系统调用请求调用的进程类型所确定的调度策略,执行针对所述引擎中实现的系统调用函数;

在调用所述系统调用函数的过程中,将所述系统调用函数的指令实时转换为对所述视窗系统内核的编程接口的调用的指令,通过调用所述视窗系统内核的编程接口实现安卓系统内核功能的调用操作,完成所述安卓应用通过所述系统调用请求所请求实现的功能;所述安卓系统内核功能是利用所述视窗系统的编程接口实现的。

2.根据权利要求1所述的方法,其特征在于,所述在视窗系统中执行所述引擎的启动操作,包括:

执行在所述视窗系统中启动对应所述引擎的窗体进程的操作,其中,所述窗体进程用于在所述视窗系统的窗体中显示所述安卓系统的运行界面、以及所述安卓应用的运行界面;

执行在所述窗体进程中启动所述引擎的常驻进程的操作。

3.根据权利要求1所述的方法,其特征在于,所述通过执行所述等效的视窗系统指令的方式运行所述安卓系统,包括:

通过执行所述等效的视窗系统指令的方式执行以下操作:

在所述引擎的常驻进程中,加载所述引擎的安卓系统初始化进程到内存;

在所述安卓系统初始化进程中,运行所述安卓系统的启动进程和系统服务。

4.根据权利要求1所述的方法,其特征在于,所述将所述安卓系统包括的安卓系统指令转换为等效的视窗系统指令,包括:

根据所述安卓系统包括的指令,查询所述安卓系统的指令集与所述视窗系统的指令集之间的转换库;

通过查询得到所述视窗系统的指令集中等效的指令。

5.根据权利要求1至4任一项所述的方法,其特征在于,所述方法还包括:

通过执行所述安卓应用的可执行文件包括的指令的方式,执行以下操作:

在所述安卓应用的用户态的进程中,通过安卓系统内核的系统调用接口,向所述引擎中实现的安卓系统内核发起所述系统调用请求,以及,

将系统调用号存储在寄存器中,所述系统调用号对应所述系统调用请求向所述安卓系统内核所请求调用的系统调用函数。

6.根据权利要求1所述的方法,其特征在于,所述方法还包括:

在所述引擎中实现的安卓系统内核中,将所述安卓应用的进程从用户态切换到内核

态，从寄存器中读取系统调用号，并根据所述系统调用号调用所述安卓系统内核中对应的系统调用函数；

在所述系统调用函数的执行过程中，调用所述系统调用函数对应的安卓系统内核功能，并将系统调用结果存储到寄存器；

将所述安卓应用的进程从内核态切换到用户态。

7.根据权利要求1至4、或6任一项所述的方法，其特征在于，所述方法还包括：

当所述系统调用请求所请求实现的功能完成时，在所述引擎中实现的安卓系统内核中，将所述安卓应用的进程从内核态返回用户态；

在所述安卓应用的用户态的进程中，从寄存器读取系统调用结果。

8.根据权利要求1至4、或6任一项所述的方法，其特征在于，所述在视窗系统中执行所述引擎的启动操作，包括：

在所述视窗系统中启动实现所述引擎的运行于用户态的进程，或者，

在所述视窗系统中启动实现所述引擎的运行于内核态的内核驱动程序。

9.根据权利要求2所述的方法，其特征在于，所述方法还包括：

在所述引擎的窗体进程中显示的所述安卓应用的界面，接收对应所述安卓应用的操作；

当响应所述操作需要使用安卓系统内核功能时，通过所述引擎中实现的安卓系统内核的系统调用接口，发起与所述安卓系统内核功能对应的所述系统调用请求。

10.根据权利要求2所述的方法，其特征在于，所述方法还包括：

当所述系统调用结果表示系统调用请求成功时，在所述窗体进程中显示所述系统调用请求的响应界面。

11.一种终端，其特征在于，设置有引擎，所述引擎包括：

进程单元，用于接收启动跨系统运行安卓应用的引擎的指令，根据所述指令在视窗系统中执行所述引擎的启动操作；视窗/安卓单元，用于在所述引擎的进程中，将所述安卓系统包括的安卓系统指令转换为等效的视窗系统指令；通过执行所述等效的视窗系统指令的方式运行所述安卓系统；解析待执行的安卓应用，得到所述安卓应用的可执行文件包括的指令；通过执行所解析到的指令的方式运行所述安卓应用；

安卓核心功能单元，用于当通过所述视窗系统内核，检测到所述安卓应用发起系统调用请求时，接收所述视窗系统的内核发送的所述系统调用请求；根据所述系统调用请求调用的进程类型所确定的调度策略，执行针对所述引擎中实现的系统调用函数；在调用所述系统调用函数的过程中，将所述系统调用函数的指令实时转换为对所述视窗系统内核的编程接口的调用的指令，通过调用所述视窗系统内核的编程接口实现安卓系统内核功能的调用操作，完成所述安卓应用通过所述系统调用请求所请求实现的功能；所述安卓系统内核功能是利用所述视窗系统的编程接口实现的。

12.根据权利要求11所述的终端，其特征在于，

所述进程单元包括：窗体进程单元，用于执行在所述视窗系统中对应所述引擎的窗体进程的操作，所述窗体进程用于在所述视窗系统的窗体中显示所述安卓系统的运行界面、以及所述安卓应用的运行界面；执行在所述窗体进程中启动所述引擎的常驻进程的操作；及在所述引擎的窗体进程中显示的所述安卓应用的界面，接收对应所述安卓应用的操作；

以及所述系统调用结果表示系统调用请求成功时，在所述窗体进程中显示所述系统调用请求的响应界面。

13.根据权利要求11所述的终端，其特征在于，

所述进程单元包括：常驻进程单元，用于通过执行所述等效的视窗系统指令的方式执行以下操作：在所述引擎的常驻进程中，加载所述引擎的安卓系统初始化进程到内存；以及在所述安卓系统初始化进程中，运行所述安卓系统的启动进程和系统服务。

14.根据权利要求11至13任一项所述的终端，其特征在于，

所述视窗/安卓单元，用于根据所述安卓系统包括的指令，查询所述安卓系统的指令集与所述视窗系统的指令集之间的转换库；通过查询得到所述视窗系统的指令集中等效的指令；及通过执行所述安卓应用的可执行文件包括的指令的方式执行以下操作：在所述安卓应用的用户态的进程中，通过安卓系统内核的系统调用接口，向所述引擎中实现的安卓系统内核发起所述系统调用请求，以及，将系统调用号存储在寄存器中，所述系统调用号对应所述系统调用请求向所述安卓系统内核所请求调用的系统调用函数；及在所述引擎中实现的安卓系统内核中，将所述安卓应用的进程从用户态切换到内核态，从寄存器中读取系统调用号，并根据所述系统调用号调用所述安卓系统内核中对应的系统调用函数；在所述系统调用函数的执行过程中，调用所述系统调用函数对应的安卓系统内核功能，并将系统调用结果存储到寄存器；将所述安卓应用的进程从内核态切换到用户态；及当所述系统调用请求所请求实现的功能完成时，将所述安卓应用的进程从内核态返回用户态时，在所述安卓应用的用户态的进程中，从寄存器读取系统调用结果；以及当响应所述操作需要使用安卓系统内核功能时，通过所述引擎中实现的安卓系统内核的系统调用接口，发起与所述安卓系统内核功能对应的所述系统调用请求；

所述进程单元，具体用于在所述视窗系统中启动实现所述引擎的运行于用户态的进程，或者，在所述视窗系统中启动实现所述引擎的运行于内核态的内核驱动程序。

15.一种计算机可读存储介质，其特征在于，存储有可执行指令，用于引起一个或多个处理器执行权利要求1至10任一项所述的跨系统运行安卓应用的方法。

# 一种跨系统运行安卓应用的方法、终端及存储介质

## 技术领域

[0001] 本发明涉及电数字数据处理技术，尤其涉及一种跨系统运行安卓应用的方法、终端及存储介质。

## 背景技术

[0002] 安卓(Android)系统由于其开源性发展迅速，各大手机厂商纷纷投入生产、设计并开发自己的安卓系统，目前安卓系统已经超越iOS系统成为全球最有影响力的系统。随着人们每天用手机消遣娱乐的时间越来越多，由于不同手机本身的局限性，以及不同手机的性能、电池、散热等方面的影响，使用手机进行应用时，往往不能达到最佳的应用性能和效果。因此，把安卓系统移植到具有更高硬件配置的设备如个人电脑(PC，Personal Computer)的视窗(Windows)系统中运行，利用高配置的个人电脑来运行安卓游戏等安卓应用的需求应运而生。

[0003] 相关技术通常采用各种虚拟化技术在视窗系统中虚拟安卓系统，从而实现在虚拟的安卓系统中运行安卓应用的目的，但是利用虚拟化技术跨系统运行安卓应用时，在兼容性和执行效率上都与在安卓系统中直接运行安卓应用存在明显的差距，同时也导致挤占了设备的大量硬件资源(例如CPU和内存)。

## 发明内容

[0004] 为解决上述技术问题，本发明实施例期望提供一种跨系统运行安卓应用的方法、终端及存储介质，能够以良好的兼容性和效率跨系统运行安卓应用。

[0005] 本发明实施例的技术方案是这样实现的：

[0006] 本发明实施例提供了一种跨系统运行安卓应用的方法，包括：

[0007] 接收启动跨系统运行安卓应用的引擎的指令，根据所述指令在视窗系统中执行所述引擎的启动操作；

[0008] 在所述引擎启动后的进程中，将所述安卓系统包括的安卓系统指令转换为等效的视窗系统指令；

[0009] 通过执行所述等效的视窗系统指令的方式运行所述安卓系统；

[0010] 解析待执行的安卓应用，得到所述安卓应用的可执行文件包括的指令；

[0011] 通过执行所解析到的指令的方式运行所述安卓应用；

[0012] 当解析到发起系统调用请求的指令时，执行针对所述引擎中实现的系统调用函数、以及实现的安卓系统内核功能的调用操作；

[0013] 其中，所述调用操作用于完成所述安卓应用通过所述系统调用请求所请求实现的功能。

[0014] 本发明实施例提供了一种终端，设置有引擎，所述引擎包括：

[0015] 进程单元，用于接收启动跨系统运行安卓应用的引擎的指令，根据所述指令在视窗系统中执行所述引擎的启动操作；

[0016] 视窗/安卓单元，用于在所述引擎的进程中，将所述安卓系统包括的安卓系统指令转换为等效的视窗系统指令；通过执行所述等效的视窗系统指令的方式运行所述安卓系统；解析待执行的安卓应用，得到所述安卓应用的可执行文件包括的指令；通过执行所解析到的指令的方式运行所述安卓应用；

[0017] 安卓核心功能单元，用于当所述视窗/安卓单元解析到发起系统调用请求的指令时，执行针对所述引擎中实现的系统调用函数、以及实现的安卓系统内核功能的调用操作；其中，所述调用操作用于完成所述安卓应用通过所述系统调用请求所请求实现的功能。

[0018] 本发明实施例提供了一种计算机可读存储介质，存储有可执行指令，用于引起一个或多个处理器执行所述的跨系统运行安卓应用的方法。

[0019] 本发明实施例具有以下有益效果：

[0020] 在引擎中执行安卓系统指令到视窗系统指令的转换，并通过执行转换得到的视窗系统指令的方式运行安卓系统，从视窗系统的指令执行的层面实现安卓系统和安卓应用，这种底层的实现方式相较于相关技术在视窗系统中虚拟安卓系统效率更高，还降低了硬件资源的要求；

[0021] 在引擎中利用自实现的系统调用和安卓系统内核来完成安卓应用的系统调用请求，由于不需要依赖于第三方库，相较于相关技术提供的虚拟化方案，克服了对第三方库的依赖，因而能够在各种版本的视窗系统中运行安卓应用，具有良好的兼容性。

## 附图说明

[0022] 图1为本发明实施例提供的示例性的系统调用的示例图；

[0023] 图2-A为本发明实施例提供的具有视窗系统的终端的内部结构示意图；

[0024] 图2-B为本发明实施例提供的具有视窗系统的终端的内部结构示意图；

[0025] 图2-C为本发明实施例提供的具有视窗系统的终端的内部结构示意图；

[0026] 图3为本发明实施例提供的引擎的模块结构示意图；

[0027] 图4为本发明实施例提供的跨系统运行安卓应用的方法的流程图；

[0028] 图5为本发明实施例提供的系统调用的示例图；

[0029] 图6为本发明实施例提供的安卓系统启动的示例图；

[0030] 图7为本发明实施例提供的引擎的结构图；

[0031] 图8为本发明实施例提供的用户态和内核态的特权权限级别分布图；

[0032] 图9为本发明实施例提供的系统调用的示例图；

[0033] 图10为本发明实施例提供的服务器与终端进行信息交互的硬件实体系统架构图；

[0034] 图11为本发明实施例提供的终端的显示界面示意图。

## 具体实施方式

[0035] 为了使本发明的目的、技术方案和优点更加清楚，下面将结合附图对本发明作进一步地详细描述，根据本发明的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其它实施例，都属于本发明保护的范围。

[0036] 除非另有定义，本文所使用的所有的技术和科学术语与属于本发明的技术领域的技术人员通常理解的含义相同。本文中所使用的术语只是为了描述具体的实施例的目的，

不是旨在限制本发明。

[0037] 对本发明进行进一步详细说明之前,对本发明实施例中涉及的名词和术语进行说明,本发明实施例中涉及的名词和术语适用于如下的解释。

[0038] 1)跨系统,本文中指安卓系统的原生应用(下文中简称为安卓应用)在区别于安卓(Android)系统的视窗系统运行,其中,安卓系统内核基于Linux内核实现,视窗系统使用微软视窗新技术(Windows NT,Microsoft Windows New Technology)内核,例如:Windows7、Windows8和Windows10等各种发行版本,提供32位平坦寻址,使用32位指令集,具有高安全性和对精简指令集(RISC,Reduced Instruction Set Computer)机器的可移植性。

[0039] 2)用户态,当安卓应用的进程在执行安卓系统自身的代码时,称进程处于用户运行态(简称为用户态),进程此时处于最低特权级别(例如在安卓系统中为ring3);运行于用户态的进程可以执行的操作和访问的资源都会受到限制,一般地,进程默认运行于用户态,进程在通常情况下不允许访问内核数据,也无法使用Linux内核功能,只能在用户空间操作用户数据,调用用户空间函数,当一些操作需要使用安卓系统的安卓系统内核功能,例如内存管理、驱动程序、驱动程序和输入/输出(I/O)管理等时,进程会从用户态切换到内核态并陷入内核代码去执行,这个切换过程称为进程陷入(Trap)内核态。

[0040] 3)内核态,当安卓应用的进程通过发起系统调用请求而陷入内核代码中执行时,称进程处于内核运行态(简称为内核态),进程此时处于最高特权级别(例如在安卓系统中为ring0),当进程处于内核态时,由Linux内核为进程在执行系统调用函数,内核态的进程可以执行任何操作并且在资源的使用上没有限制。

[0041] 4)系统调用请求,安卓应用中用于调用系统调用接口的指令,也称为系统调用指令,例如图1中用户态的进程执行了调用系统调用接口abc()的指令,安卓应用的进程在陷入内核态之前,会通过向寄存器存储系统调用函数的参数(包括系统调用号)的方式,向内核态以声明需要调用的系统调用函数,例如图1中用户态的进程执行了调用系统调用接口abc()的指令之前,会将eax寄存器的值置为需要调用的系统调用函数对应的系统调用号。

[0042] 5)系统调用接口(System Call Interface),安卓系统内核中开放给安卓应用发起系统调用请求的统一入口,通过向寄存器存入系统调用函数的参数(例如系统调用号),并定义返回值在寄存器的存储位置,通过特定软件中断指令使用户态的进程陷入内核态,跳转到Linux内核的系统调用处理程序的入口执行,在系统调用处理程序结束后将进程返回用户态。仍以图1为例,系统调用接口abc()使用int 0x80中断指令而使进程陷入内核态,跳转到系统调用处理程序的入口(记为system_call)执行,在系统调用处理程序结束后返回。

[0043] 6)系统调用(System Call)函数,简称为系统调用,是进程进入内核的接口层,系统调用是由内核函数(实现安卓系统内核功能的一系列函数,诸如进程调度、内存管理、文件系统、渲染和网络的函数实现)实现,实现系统调用的内核函数被称为系统调用服务例程,因此系统调用也可以称为系统调用服务例程的封装例程。

[0044] 7)系统调用处理程序,也称为系统调用例程,是int 0x80中断指令对应的中断处理程序,当进程通过系统调用接口发出int 0x80中断指令时,系统调用处理程序进行现场保护,例如将寄存器的内容存入内核堆栈,通过寄存器传入内核态的参数,以及根据系统调用表(System Call Table)存储的系统调用号(System Call Number)与系统调用函数的指

针的对应关系，跳转到相应的系统调用函数执行，并将参数传递给系统调用服务例程，在系统调用服务例程结束后，利用内核堆栈的内容恢复现场，将从系统调用服务例程获得的调用结果(例如，调动成功或者出错条件)存入寄存器，然后通过执行sysret/iret指令由中断返回系统调用接口，从而将进程从内核态切换回用户态。

[0045] 8)Windows系统中的安卓系统(AOW，Android on Windows)引擎，包括一套在引擎中自实现的Linux内核，以及安卓系统与Windows系统之间的实时指令转换功能，用于支持在在Windows系统中运行安卓应用。

[0046] 9)应用程序接口(API，Application Program Interface)，即预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。

[0047] 下面说明实现本发明实施例的跨系统运行安卓应用的方法的引擎的示例性应用，结合引擎在视窗系统中的设置进行示例说明。

[0048] 如图2-A所示，图2-A为本发明实施例提供的运行视窗系统的终端的内部结构示意图，在终端2中，包括：应用层20、操作系统层21和硬件层22。操作系统层21运行视窗系统。应用层20包括视窗系统的原生应用即视窗系统应用201，通过在视窗系统中设置的引擎还可以运行安卓系统应用200。硬件层22包括处理器220和存储器221。

[0049] 下面结合图2-B和图2-C对在视窗系统中设置引擎进行示例性说明。

[0050] 在本发明一些实施例中，如图2-B所示，本发明实施例提供的实现跨系统运行安卓应用的方法的引擎1可以实施为终端2的应用层20的应用，例如用户在终端2的操作系统21中安装引擎1的软件包的方式，使终端2运行实现引擎1的应用程序，在应用程序的进程中实现安卓系统的内核功能。

[0051] 在本发明一些实施例中，如图2-C所示，引擎1可以实施为在操作系统层21的内核驱动程序，通过在操作系统层21安装实现引擎1的内核驱动程序，在内核驱动程序的进程中实现安卓系统的内核功能。

[0052] 在本发明的一些实施例中，终端2可以包括：处理器220和存储器221。存储器221，用于存储可执行指令；处理器220，用于执行所述存储器存储的可执行指令时，实现本发明实施例提供的跨系统运行安卓应用的方法。

[0053] 实现本发明实施例的终端为运行有视窗系统并根据上述任一方式设置有引擎的电子设备，例如台式机电脑和平板电脑等，本发明实施例不作限制。

[0054] 这里，处理器220可以为中央处理器(CPU，Central Processing Unit)、微处理器(MPU，Microprocessor Unit)、数字信号处理器(DSP，Digital Signal Processing)或现场可编程门阵列(FPGA，Field Programmable Gate Array)等。

[0055] 需要说明的是，实际应用时，终端2中的各个组件通过通信总线耦合在一起。可理解，通信总线用于实现这些组件之间的连接通信。通信总线除包括数据总线之外，还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见，将各种总线都标为通信总线。

[0056] 可以理解，存储器221可以是高速RAM存储器，也可以是非不稳定的存储器(Non-Volatile Memory)，例如至少一个磁盘存储器。存储器221还可以是至少一个远离处理器220的存储系统。

[0057] 本发明实施例提供的跨系统运行安卓应用的方法由处理器220执行可执行指令实

现。处理器220可能是一种集成电路芯片，具有信号的处理能力。在实现过程中，应用于跨系统运行安卓应用的方法中的不同操作可以通过软件形式的指令完成。上述的处理器220可以是通用处理器、DSP或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等。处理器220可以实现跨系统运行安卓应用的方法、步骤及逻辑框图。通用处理器可以是微处理器或者任何常规的处理器等。

[0058] 在本发明的一些实施例中，引擎1实现的跨系统运行安卓应用的方法是通过软件形式的指令完成。

[0059] 作为示例，实现引擎1的可执行指令可以存储在存储介质中，存储介质位于存储器221，处理器220读取存储器221中的信息，结合其硬件完成本发明实施例提供的跨系统运行安卓应用的方法，如图3所示，引擎1包括进程单元10、视窗/安卓单元11和安卓核心功能单元12。其中，进程单元10中包括：窗体进程单元100和常驻进程单元101。

[0060] 进程单元10，用于接收启动跨系统运行安卓应用的引擎的指令，根据所述指令在视窗系统中执行所述引擎的启动操作；

[0061] 视窗/安卓单元11，用于在所述引擎的进程中，将所述安卓系统包括的安卓系统指令转换为等效的视窗系统指令；通过执行所述等效的视窗系统指令的方式运行所述安卓系统；解析待执行的安卓应用，得到所述安卓应用的可执行文件包括的指令；通过执行所解析到的指令的方式运行所述安卓应用；

[0062] 安卓核心功能单元12，用于当所述视窗/安卓单元解析到发起系统调用请求的指令时，执行针对所述引擎中实现的系统调用函数、以及实现的安卓系统内核功能的调用操作；其中，所述调用操作用于完成所述安卓应用通过所述系统调用请求所请求实现的功能。

[0063] 在一些实施例中，所述进程单元10包括：窗体进程单元100，用于执行在所述视窗系统中对应所述引擎的窗体进程的操作，所述窗体进程用于在所述视窗系统的窗体中显示所述安卓系统的运行界面、以及所述安卓应用的运行界面；执行在所述窗体进程中启动所述引擎的常驻进程的操作。

[0064] 在一些实施例中，所述进程单元10包括：常驻进程单元101，用于通过执行所述等效的视窗系统指令的方式执行以下操作：在所述引擎的常驻进程中，加载所述引擎的安卓系统初始化进程到内存；以及在所述安卓系统初始化进程中，运行所述安卓系统的启动进程和系统服务。

[0065] 在一些实施例中，所述视窗/安卓单元11，具体用于根据所述安卓系统包括的指令，查询所述安卓系统的指令集与所述视窗系统的指令集之间的转换库；通过查询得到所述视窗系统的指令集中等效的指令。

[0066] 在一些实施例中，所述视窗/安卓单元11，还用于通过执行所述安卓应用的可执行文件包括的指令的方式执行以下操作：在所述安卓应用的用户态的进程中，通过安卓系统内核的系统调用接口，向所述引擎中实现的安卓系统内核发起所述系统调用请求，以及，将系统调用号存储在寄存器中，所述系统调用号对应所述系统调用请求向所述安卓系统内核所请求调用的系统调用函数。

[0067] 在一些实施例中，所述视窗/安卓单元11，具体用于在所述引擎中实现的安卓系统内核中，将所述安卓应用的进程从用户态切换到内核态，从寄存器中读取系统调用号，并根据所述系统调用号调用所述安卓系统内核中对应的系统调用函数；在所述系统调用函数的

执行过程中，调用所述系统调用函数对应的安卓系统内核功能，并将系统调用结果存储到寄存器；将所述安卓应用的进程从内核态切换到用户态。

[0068] 在一些实施例中，所述视窗/安卓单元11，还用于当所述系统调用请求所请求实现的功能完成时，将所述安卓应用的进程从内核态返回用户态时，在所述安卓应用的用户态的进程中，从寄存器读取系统调用结果。

[0069] 在一些实施例中，所述进程单元10，具体用于在所述视窗系统中启动实现所述引擎的运行于用户态的进程，或者，在所述视窗系统中启动实现所述引擎的运行于内核态的内核驱动程序。

[0070] 在一些实施例中，所述进程单元10包括：窗体进程单元100，用于在所述引擎的窗体进程中显示的所述安卓应用的界面，接收对应所述安卓应用的操作；

[0071] 所述视窗/安卓单元11，还用于当响应所述操作需要使用安卓系统内核功能时，通过所述引擎中实现的安卓系统内核的系统调用接口，发起与所述安卓系统内核功能对应的所述系统调用请求。

[0072] 在一些实施例中，所述进程单元10包括：窗体进程单元100，用于所述系统调用结果表示系统调用请求成功时，在所述窗体进程中显示所述系统调用请求的响应界面。

[0073] 至此，已经说明了本发明实施例提供的引擎的各种示例性实施。

[0074] 在本发明一些实施例中，提供基于虚拟化技术的Android模拟器运行安卓应用的方案，Android模拟器分为两种基本类型：Bluestacks和Virtualbox。Bluestacks是最早在PC上实现流畅运行安卓系统的方案。Bluestacks的原理是开发了一套虚拟机机制，由于是独立开发的虚拟机，所以对PC硬件本身没有要求，在硬件兼容性方面有一定优势。Virtualbox是Oracle公司开发，开源的虚拟机方案，通过在Windows内核底层直接插入驱动模块，创建一个完整虚拟的电脑环境运行安卓系统，由虚拟化技术构造出安卓系统需要的运行系统，用户可在虚拟机中运行对应的Android游戏。

[0075] 然而，Bluestacks需要模拟的Android接口数量巨大，很难面面俱到，而且虚拟机的运行过程本身需要耗费PC端的大量硬盘空间和内存，PC端大量的系统资源被占用，将导致降低PC端各进程运行的响应速度。而Virtualbox通过CPU VT硬件加速，性能和兼容性都会更好，但是必须安装该虚拟机的全套内核级的驱动程序，在应用程序的运行过程中调用各内核驱动的过程极为复杂，且耗费系统资源，对电脑配置要求较高，而且需要PC有一定的配置条件，适用性受到限制；可见，利用Android模拟器运行安卓应用的方案，存在兼容性和效率的问题。

[0076] 针对上述问题，继续结合图1至图3，对实现本发明实施例的跨系统运行安卓应用的方法的示例性实施进行说明。本发明实施例提供了一种跨系统运行安卓应用的方法，如图4所示，涉及步骤S101至步骤S107，下面将对各个步骤进行说明。

[0077] 步骤S101、接收启动跨系统运行安卓应用的引擎的指令，根据指令在视窗系统中执行引擎的启动操作。

[0078] 本发明实施例提供的在视窗系统中启动的跨系统运行安卓系统的引擎简称为AOW引擎，在对启动AOW引擎进行说明之前，首先说明实现本发明实施例的AOW引擎。

[0079] 在基于Windows NT内核的各种版本的Windows系统中运行安卓系统以及安卓应用，即跨系统运行安卓应用，其中，视窗系统作为安卓系统的宿主系统，作为示例，运行

Windows系统的电子设备可以是台式电脑、笔记本电脑或者平板电脑等各种终端设备，本发明实施例不作限制。

[0080] 在本发明一些实施例中，安卓系统中的操作将涉及到Linux内核的相关操作，如Linux内核的数据结构，信号管理机制等，在AOW引擎中进行了深度定制，在AOW引擎中利用Windows底层API和Windows NT内核实现Linux内核，包括Linux内核的系统调用(函数)和内核功能模块(例如，在Windows系统用户层或内核层实现)，如：进程调度、内存管理、文件系统等，从而支持在Windows系统中运行安卓系统，进而可以在安卓系统中运行安卓应用。

[0081] 举例来说，实现本发明实施例的AOW引擎是一套组件，使得安卓系统环境中的可执行格式(ELF，Executable and Linking Format)的二进制文件(下文中简称为二进制文件)，从而使安卓系统以及安卓应用的二进制文件能够运行于Windows系统中。

[0082] 下面对Windows系统特权权限进行说明，以Windows系统的硬件平台为Intel的386CPU为例，存在4个特权权限：从ring0到ring3；其中，ring0的特权权限级别最高，任意代码可以在用户态运行，ring3特权权限级别最低，只能执行有限代码；在Windows系统中，内核态的特权权限级别为ring0，用户态的特权权限级别为ring3，Windows系统的核心代码运行在内核态以避免受到恶意的攻击；而运行在用户态的应用程序的特权权限因受到限制可以保证系统安全，如果应用程序进行一些诸如直接访问物理内存的动作，那么就需要切换到内核态去执行Windows系统的核心代码来完成所需的功能，这样可以保证系统安全。

[0083] 举例来说，Windows系统中的应用程序处于用户态时，应用程序需要切换为内核态以调用Windows系统的子系统中的API，这些子系统除了Win32，还有POSIX、OS/2等，这里，为兼容16位应用程序或其他系统应用程序，需要通过调用Win32子系统来实现。Win32子系统将API函数转化为底层API，系统版本的变化只是底层API的改变，并没有修改Win32，即没有改变Win32API函数的名称和参数，这样实现了对应用程序的兼容，在底层API中的函数调用被转化为系统服务函数调用并进入内核态，并进一步向下传递实现相应功能。

[0084] 在本发明一些实施例中，Windows系统的Windows NT内核明确地分离了用户态和内核态，所有的应用程序再也不可能无意之中改写系统进程使用的数据和进程空间的内容，从而避免系统出现人为的非法操作，令Windows系统更安全，稳定。

[0085] 根据上述说明Windows系统中的用户态和内核态，AOW引擎在Windows系统中可以实现为用户态，以保证Windows系统的安全性；当然AOW引擎在Windows系统中也可以实现为内核态，下面就内核态AOW引擎(在硬件层实现引擎的情况)和用户态AOW引擎(在应用层实现引擎的过程)两种版本进行说明。

[0086] 作为示例，AOW引擎在Windows系统的用户态实现(即用户态版本)，用户态版本的AOW引擎实现为Windows系统应用程序，利用Windows底层的API的编程，实现Linux内核的系统调用(函数)和Linux内核功能(是内核函数的封装)，从而实现Linux的系统环境。

[0087] 作为示例，AOW引擎在Windows系统的内核态实现(即内核态版本)，内核态版本的AOW引擎实现为Windows系统中内核驱动程序，把Linux内核的系统调用(函数)和Linux内核的功能在Windows NT内核的驱动层实现。

[0088] 根据上述示例，本发明实施例提供的AOW引擎相当于在安卓系统的二进制文件和Windows NT内核之间的一层介质，通过AOW引擎加载安卓系统环境中可执行文件，并运行其中的针对Linux内核的系统调用，并利用在AOW引擎中实现Linux内核完成系统调用请求。

[0089] 在本发明一些实施例中，由于Windows NT内核允许运行子系统环境，所以可以利用Windows NT内核提供的API的编程功能，根据实际需求在AOW引擎中开发、扩展Windows系统中用于运行安卓应用的子系统，这样，AOW引擎就可以在Windows系统中跨系统运行安卓系统以及安卓应用。

[0090] 至此，已经说明了实现本发明实施例的AOW引擎，继续说明AOW引擎在Windows系统中的启动过程。

[0091] 在本发明一些实施例中，在引擎所在的终端开机时，引擎会接收启动跨系统运行安卓应用的引擎的指令；根据上述指令就可以在视窗系统中执行引擎的启动操作，实现引擎的启动。

[0092] 在本发明一些实施例中，根据上述指令在视窗系统中执行引擎的启动操作时，执行在Windows系统中启动对应AOW引擎的窗体进程的操作，使得窗体进程先启动，其中，窗体进程用于在视窗系统的窗体中显示安卓系统的运行界面、以及安卓应用的运行界面，然后执行在窗体进程中启动Windows系统中对应AOW引擎的常驻进程的操作。针对用户态版本的AOW引擎，AOW引擎的窗体进程启动后，执行在视窗系统中启动实现引擎的运行于用户态的进程的操作，即在窗体进程中启动Windows系统中对应AOW引擎的Native进程，即引擎的Native进程。针对内核态版本的AOW引擎，AOW引擎的窗体进程启动后，执行在视窗系统中启动实现引擎的运行于内核态的内核驱动程序的驱动进程的操作，从而启动AOW引擎。

[0093] 作为示例，假设AOW引擎的窗体进程为aow_gui进程，当AOW引擎启动时，AOW引擎的aow_gui进程首先启动，aow_gui进程是一个Win32的与用户交互的窗体程序，aow_gui进程启动后，针对用户态版本的AOW引擎，会开启Windows系统中对应AOW引擎的Native进程，在Native进程会启动AOW引擎的核心功能。针对内核态版本的AOW引擎，在视窗系统中启动实现引擎的运行于内核态的内核驱动程序，以启动AOW引擎的核心功能，也就是说，内核态实现的AOW引擎就是写在驱动程序中的内核功能代码实现的。

[0094] 在本发明一些实施例中，AOW引擎实现的Linux内核的核心功能可以包括：进程调度、内存管理、文件系统、网络和渲染。

[0095] 在本发明一些实施例中，利用Windows NT内核的进程调度机制和进程间通信的方式来实现Linux内核的进程管理方案，在AOW引擎用到了管道机制、信号机制、共享内存机制和Socket机制。

[0096] 在本发明一些实施例中，可以将基于Linux文件系统的安卓相关数据的数据结构转换为基于Windows系统的相关数据的数据结构，以实现数据在安卓系统及Windows系统之间的正常便捷通信，在Windows系统中建立并处理安卓系统的相应信号、并在Windows系统中管理安卓系统的内存分配等。下面针对内存管理、文件系统、网络和渲染再进行详细的说明。

[0097] 内存管理：安卓系统和Windows系统的内存管理方式是不同的，AOW引擎基于Windows NT内核API中的虚拟内存管理机制(VirtualAlloc、CreateFileMapping等)实现了一套兼容Linux中mmap2、brk等系统调用的内存管理机制。

[0098] 文件系统：AOW引擎提供了ext4文件系统支持，包含了安卓系统的根目录和应用程序文件(/data，/system，/etc等)，利用Windows NT内核的原生进程文件读写机制(NtCreateFile、NtReadFile、NtWriteFile等)实现了一套兼容ext4文件系统的机制。

[0099] 网络:AOW引擎会从Windows系统查询网络接口列表和DNS服务器,并将这些信息传递Linux内核的核心功能模块,之后会在本地缓存有关各种网络接口的所有信息。AOW引擎将通过调用已注册的回调处理程序来通知服务对被监视网络信息的任何更改。当有信息更改时,回调处理程序会将信息通知给Linux内核的核心功能模块。在AOW引擎实现的Linux内核里对网络请求进行封装处理,然后通过进程间通信方式-管道技术把封装的包传输到aow_gui进程解包,转换成Windows系统中可识别的形式。

[0100] 渲染:openGL是免费的2D/3D图形API软件库,是一套绘图函数的标准,在安卓上应用的是一个分支子集openGLES,是openGL针对嵌入式设备而设计的。AOW引擎中安卓系统调用底层自实现的openGLES接口,把函数名、参数封装成数据流,然后通过管道传给aow_gui进程,由aow_gui进程解包出函数名、参数等转换成Windows系统可以识别的openGL或directx接口函数。

[0101] 步骤S102、在引擎启动后的进程中,将安卓系统包括的安卓系统指令转换为等效的视窗系统指令。

[0102] 步骤S103、通过执行等效的视窗系统指令的方式运行安卓系统。

[0103] 步骤S104、解析待执行的安卓应用,得到安卓应用的可执行文件包括的指令。

[0104] 步骤S105、通过执行所解析到的指令的方式运行安卓应用。

[0105] 步骤S106、当解析到发起系统调用请求的指令时,执行针对引擎中实现的系统调用函数、以及实现的安卓系统内核功能的调用操作;其中,调用操作用于完成安卓应用通过系统调用请求所请求实现的功能。

[0106] 在本发明一些实施例中,在引擎启动后的进程中,将安卓系统包括的安卓系统指令转换为等效的视窗系统指令,这样,通过执行等效的视窗系统指令的方式运行安卓系统,从而在安卓系统中实现对安卓应用的运行和应用。

[0107] 在本发明一些实施例中,对于转换Android系统指令为Windows系统指令而言,通过如下方式实现:根据安卓系统包括的指令,查询安卓系统的指令集与视窗系统的指令集之间的转换库;通过查询得到视窗系统的指令集中等效的指令。即在AOW引擎的常驻进程中,根据Android系统的指令集与Windows系统的指令集之间的转换库,查询出与Android系统包括的指令、以及Android应用包括的指令对应的Windows系统的指令集中等效的指令。

[0108] 举例来说,Windows系统的平台为x86平台,使用x86架构的CPU,而Android系统的平台为高级精简指令集机器(ARM,Advanced RISC Machine)平台,Android系统以及Android应用的二进制文件中包括一系列ARM指令,这些ARM指令是根据ARM平台而编译的,而在Windows系统中只能运行针对x86平台而编译的指令即x86指令,ARM指令无法在基于x86平台的Windows系统中执行,通过ARM-x86库将ARM指令转换为等效的x86指令,通过在Windows系统中执行转换后得到的x86指令,Android系统和Android应用得以在Windows系统中运行。

[0109] 在本发明一些实施例中,对于通过执行Windows系统指令运行Android系统而言,可以通过如下方式实现:在引擎的常驻进程中,加载引擎的安卓系统初始化进程到内存;在安卓系统初始化进程中,运行安卓系统的启动进程和系统服务。即在AOW引擎对应的常驻进程中,加载AOW引擎的Android系统初始化进程到内存,在Android系统初始化进程中运行Android系统的启动进程和系统服务,从而实现Android系统的启动。

[0110] 作为示例,安卓系统的初始化进程为init进程,在安卓系统中,Linux内核启动的第一个用户态进程是init进程,其中,init进程为系统服务和其他启动进程的父进程。

[0111] 作为示例,AOW引擎的Windows系统驱动文件(.sys)加载到内存后,AOW引擎得以启动,AOW引擎的常驻进程会进行上述指令转换的操作,从而对安卓系统的init进程的二进制文件进行解析,将安卓系统包括的ARM指令转换为X86指令并加载进内存执行,init进程顺利执行后,开启安卓系统后续的启动进程和系统服务,从而完成安卓系统的启动。

[0112] 在本发明一些实施例中,AOW引擎的核心功能模块启动后,会把安卓根目录下的init文件加载进内存并执行,由init进程启动后续的启动进程和系统服务,从而启动安卓系统;举例来说,启动AOW引擎的核心功能(即实现的Linux内核)后,在AOW引擎的常驻进程中,将Linux/安卓的init进程加载进内存,init进程执行后会按照安卓系统的启动流程完成初始化,包括开启ZyGote、system server等进程和系统服务,由ZyGote进程启动安卓Runtime即Dalvik虚拟机,从而完成安卓系统的启动。

[0113] 其中,init进程是Linux内核启动的第一个用户态的进程,init进程会解析Linux的脚本文件init.rc,根据这个文件的内容init进程会装载安卓的文件系统,创建系统目录,初始化属性系统,启动安卓系统的重要的守护进程等,下面对Zygote进程和SystemServer进程进行介绍。

[0114] Zygote进程:当init进程初始化结束的时候,会启动Zygote进程,Zygote进程fork出应用进程,它是所有进程的父进程,Zygote进程初始化的时候会创建Dalivik虚拟机,预装载系统的资源文件和java类,所有从Zygote进程fork出的用户进程将继承和共享这些资源,不用浪费时间重新加载,创建Dalivik虚拟机完成之后,Zygote进程也将变为守护进程,负责响应启动安卓应用的启动请求。

[0115] SystemServer进程:SystemServer进程是Zygote进程fork出的第一个进程,也是整个安卓系统的核心进程,在SystemServer主要运行的是Binder服务,它首要启动本地服务SensorService,接着启动ActivityManagerService,Windows ManagerService,PackgeManagerService等在内的所有java服务。

[0116] 作为示例,如图6所示,AOW引擎启动时,启动的第一个进程是aow_gui进程,该进程是一个Win32的进程,Win32的进程是与用户交互的窗体程序,aow_gui进程会开启在Windows系统中对应的引擎常驻进程,常驻进程会在Windows驱动内核(NT Kernel)中启动AOW引擎用以实现Linux内核的核心功能:在常驻进程中将用于实现Linux内核/安卓系统实例(Instance)的init进程加载进内存,init进程执行后会按照安卓的启动流程,开启ZyGote、system server等进程和系统服务,由ZyGote进程启动安卓Runtime即Dalvik虚拟机,从而把安卓系统启动起来。其中,Windows系统中还需要通过NT PROSS先把自己的驱动加载好才可以实现上述安卓系统的启动。需要指出,Windows系统在启动安卓系统的同时,也可以根据需要会启动Windows系统自身的系统进程,例如NT PROCESS,实现Windows系统的启动。

[0117] 在本发明一些实施例中,安卓系统启动后,在AOW引擎的文件系统进程(记为aow_fs)把安卓系统根目录的所有子目录和文件转换成Linux文件系统;举例来说,安卓系统启动后,会不断读写生成的安卓系统中可以识别的Linux文件系统,Linux文件系统是安卓系统中数据的集合,Linux文件系统不仅包含着文件中的数据而且还有文件系统的结构,所有

安卓系统的用户和应用程序的文件、目录、软连接及文件保护信息等;Linux文件系统存储在安卓系统的根目录(记为rootfs)中,即启动安卓系统后生成的Linux文件系统保存在安卓系统的根目录中。

[0118] 在本发明一些实施例中,AOW引擎在启动了安卓系统之后,AOW引擎可以运行安卓系统中的安卓应用了,通过解析待执行的安卓应用,得到安卓应用的可执行文件包括的指令,最后通过执行所解析到的指令的方式运行安卓应用;当解析到发起系统调用请求的指令时,执行针对引擎中实现的系统调用函数、以及实现的安卓系统内核功能的调用操作;其中,调用操作用于完成安卓应用通过系统调用请求所请求实现的功能。

[0119] 在本发明一些实施例中,通过执行安卓应用的可执行文件包括的指令的方式,执行以下操作:在安卓应用的用户态的进程中,通过安卓系统内核的系统调用接口,向引擎中实现的安卓系统内核发起系统调用请求,以及,将系统调用号存储在寄存器中,系统调用号对应系统调用请求向安卓系统内核所请求调用的系统调用函数。

[0120] 安卓系统中运行的安卓应用的运行过程中需要使用Linux内核功能时,会发起相应功能的系统调用请求,AOW引擎会接到安卓应用的系统调用请求,由于本发明实施例中采用的用户态版本和内核态版本实现了安卓系统的系统调用和安卓系统的内核功能,AOW引擎响应于安卓系统中安卓应用运行时发起的系统调用请求,利用AOW引擎中实现的系统调用函数,以及实现的安卓系统内核功能完成系统调用请求,从而得到系统调用结果,并向安卓应用返回该系统调用结果。

[0121] 作为示例,AOW引擎利用Windows API编程实现(在用户态实现的应用程序,或者内核态层实现的内核驱动程序)了Linux系统调用(函数)和实现Linux内核。通过安卓应用在发起Linux系统调用请求时陷入实现的Linux内核,通过调用系统调用函数的方式(系统调用函数需要结合Linux内核函数完成系统调用)执行Linux内核的功能,比如:进程调度、内存管理、文件系统、网络和渲染等。此外,x86平台与ARM平台的指令集的差异性,要想在Windows系统运行安卓系统,需要进行ARM平台到x86平台的指令转换。

[0122] 作为示例,先说明AOW引擎的结构,如图7所示的AOW引擎的结构:AOW引擎涉及在用户态运行的部分以及在内核态运行的部分。其中:

[0123] 用户态:包括aow_gui进程,它是Win32的窗体程序;常驻进程,负责启动AOW引擎的核心功能;Linux/安卓(instance)主要是通Windows API编程的方式实现Linux内核的系统调用,从而可以执行Linux二进制文件。

[0124] 内核态:用户态的进程通过系统调用之后,通过Windows的DeviceIoControlAPI发送控制代码到Windows驱动程序实现Linux内核,在实现的Linux内核层完成系统调用,实现Linux内核的功能。

[0125] 其中,如图8所示,当安卓应用的进程在执行自己的代码时,称进程处于用户运行态(简称为用户态),进程此时处于最低特权级别(例如在安卓系统中为ring3);运行于用户态的进程可以执行的操作和访问的资源都会受到限制,一般地,进程默认运行于用户态,进程在通常情况下不允许访问内核数据,也无法使用Linux内核的功能,只能在用户空间操作用户数据,调用用户空间函数,当一些操作需要使用安卓系统的安卓系统内核功能,例如内存管理、驱动程序、驱动程序和输入/输出(I/O)管理等时,进程会从用户态切换到内核态,这个切换过程称为进程陷入(Trap)内核态。当安卓应用的进程通过发起系统调用请求而陷

入内核代码中执行时，称进程处于内核运行态(简称为内核态)，进程此时处于最高特权级别(例如在安卓系统中为ring0)，当进程处于内核态时，由安卓系统内核为进程在执行系统调用函数，内核态的进程可以执行任何操作并且在资源的使用上没有限制。

[0126] 基于图7所示的AOW引擎的结构，在安卓应用的用户态的进程中发起系统调用可以采用这样的方式：通过安卓系统内核的系统调用接口向AOW引擎实现的Linux内核发起系统调用请求，并将系统调用号存储在寄存器中，系统调用号对应系统调用请求所请求向安卓系统内核调用的系统调用函数。

[0127] 本发明实施例提供的AOW引擎中实现了Linux内核的诸多系统调用，在AOW引擎中把特定的中断号与AOW引擎中自实现(实现)的系统调用处理程序相关联，安卓应用在用户态可以特定中断号陷入Linux内核，并通过向寄存器传入的系统调用号，在系统调用处理程序调用对应的系统调用函数来完成系统调用。

[0128] 作为示例，系统调用请求是指安卓应用中用于调用系统调用接口的指令，也称为系统调用指令，例如图1中用户态的进程执行了调用系统调用接口abc()的指令，安卓应用的进程在陷入内核态之前，会通过向寄存器存储系统调用函数的参数(包括系统调用号)的方式，向内核态以声明需要调用的系统调用函数，例如图1中用户态的进程执行了调用系统调用接口abc()的指令之前，会将eax寄存器的值置为需要调用的系统调用函数对应的系统调用号。然后系统调用接口abc()使用int 0x80中断指令而使进程陷入内核态，跳转到系统调用处理程序的入口(记为system_call)执行调用系统调用函数对应的安卓系统内核功能，得到系统调用结果，并向寄存器存储系统调用结果，最后在系统调用处理程序结束后从内核态返回用户态。即调用引擎中实现的系统调用函数，以及实现的安卓系统内核功能的过程为：在AOW引擎实现安卓系统的内核(系统调用处理程序)中，将安卓应用的进程从用户态切换到内核态，从寄存器中读取系统调用号，并根据系统调用号调用安卓系统内核中对应的系统调用函数；在系统调用函数的执行过程中，调用系统调用函数对应的安卓系统内核功能，得到系统调用结果，并将系统调用结果存储到寄存器中；最后，将安卓应用的进程从内核态切换到用户态。这样，当系统调用请求所请求实现的功能完成时，在引擎中实现的安卓系统内核(系统调用处理程序)中，将安卓应用的进程从内核态返回用户态；在安卓应用的用户态的进程中，从寄存器读取系统调用结果。即AOW引擎可以响应于在安卓系统的系统调用处理程序中将安卓应用的进程从内核态返回用户态；在安卓应用的用户态的进程中，读取寄存器中的系统调用结果。

[0129] 继续对AOW引擎完成系统调用的过程进行说明，AOW引擎实现了系统调用处理程序，也称为系统调用例程，是int 0x80中断指令对应的中断处理程序，当进程通过系统调用接口发出int 0x80中断指令时，系统调用处理程序进行现场保护，例如将寄存器的内容存入内核堆栈，根据进程通过寄存器传入内核态的参数，以及系统调用表(System Call Table)存储的系统调用号(System Call Number)与系统调用函数的指针的对应关系，跳转到相应的系统调用函数执行，并将参数传递给系统调用服务例程，在系统调用服务例程结束后，利用内核堆栈的内容恢复现场，将从系统调用服务例程获得的调用结果(例如，调动成功或者出错条件)并存入寄存器，然后通过执行sysret/iret指令由中断返回系统调用接口，从而将进程从内核态切换回用户态。

[0130] 在安卓系统中，进程通过系统调用的方式陷入Linux内核，具体的实现形式是通过

软中断模拟INT 80h中断来完成,AOW引擎利用Windows API实现Linux内核的系统调用,从而可以执行Linux ELF二进制文件,进入Linux内核前进程会把相应的参数(即系统调用函数的相关参数,包括系统调用号)也(通过存储寄存器的方式)传入到Linux内核,根据参数,选择特定的Linux内核功能实现,也就是调用相应的系统调用函数实现,其中系统调用函数依赖于所对应的服务例程(也就是Linux内核的功能)实现;这实现的过程也就是对系统调用函数进行调用的相关指令,转换为对Windows NT内核的API进行调用的相关指令的过程。

[0131] 如图5所示,在安卓系统中,当安卓应用的进程执行系统调用时,会执行调用系统调用库中定义的某个调用函数的系统调用请求,该系统调用函数通常会被展开成_syscallIN()的形式,用于系统调用的格式转换和参数的传递,N为系统调用的参数的个数;进程执行到安卓应用的系统调用请求时,实际上是执行了由宏定义syscallN展开的函数,通过INT 0x80h来陷入Linux内核,同时其参数也将通过寄存器传往Linux内核。AOW引擎利用Windows API实现Linux内核的系统调用,从而可以执行Linux ELF二进制文件,当执行相应的系统调用时,进程会把相应的参数也传入到Linux内核,根据不同的参数,选择特定的Linux内核的功能实现。AOW引擎实现了Linux内核的诸多系统调用,把中断号与自实现的系统调用相关联,通过传入的调用号不同,调用不同的处理函数,本发明实施例中的内核功能可以为进程调度、内存管理和文件系统等功能的实现等,本发明实施例不作限制。

[0132] 在本发明一些实施例中,AOW引擎通过Windows NT内核API编程功能,实现Linux内核来执行Linux ELF二进制文件。AOW引擎负责处理与Windows NT内核协调兼容的Linux内核的系统调用请求,它不包含Linux的内核代码,而是一套与Linux兼容的内核。在AOW引擎上,当系统调用由安卓应用的二进制文件发出时,AOW引擎会通过Windows NT内核API编程功能,实现Linux内核来执行Linux内核的系统调用。

[0133] 作为示例,如图9所示,以getdents为例,当发生系统调用请求时,通过INT 80h中断陷入内核态,Windows NT内核检测到系统调用请求来自于AOW的进程(例如Linux的用户态操作程序Is),会把该系统调用请求发送给AOW引擎中用户态版本Native进程或内核态版本的驱动程序,Native进程或驱动进程(常驻进程)通过寄存器中的参数决定去调用哪个系统调用(函数),在调动这个系统调用函数的过程中进行指令的实时转换,将对系统调用函数的调用,转换为对Windows NT内核的API调用的指令的转换,完成系统调用,返回Windows NT内核,更新寄存器中的参数(例如将系统调用结果存储在寄存器中),并调sysret/iretq指令返回到用户态。在Windows NT内核检测到系统调用请求来自于Windows进程(Windows用户态操作程序,例如dir)时,直接去Windows NT内核的驱动程序中实现Windows系统调用即可。

[0134] 基于图9,当需要查找安卓系统的安卓文件系统中目录下的文件时,安卓系统可以通过Is命令实现文件管理功能(例如查找文件功能),而Windows系统可以通过dir命令实现文件管理功能。举例来说,在Windows系统中,需要查找安卓系统中的特定文件时,通过执行Is命令,在安卓文件系统中实现若干个系统调用来实现文件查找的过程。

[0135] 在本发明一些实施例中,本发明实施例中的指令转换指的是ARM-x86指令转换。ARM-x86指令转换体现在AOW引擎的整个运行过程中,AOW引擎的目的是要在Windows系统运行安卓系统,安卓系统的应用程序(apk、dex、odex)是被Dalvik(AOW引擎用的安卓4.4.2)解析,而Dalvik专门对于ARM平台做了优化和兼容,最终为了实现在Windows系统运行安卓应

用程序，需要使用ARM-x86库做指令转换，把ARM指令翻译为x86指令。AOW引擎在Windows的常驻进程中，负责把安卓的init进程加载并执行，从而将整个安卓系统启动；在启动后的Linux内核的系统调用实现、以及将实现出的Linux内核的核心功能模块转换为Windows NT等效的核心功能模块；上述的过程都是需要ARM实时的进行指令转换为x86指令的操作。

[0136] 在本发明一些实施例中，系统调用的类型会有很多种，对应很多种不同的功能实现，而针对不同的进程，会采用不同的调度策略。

[0137] 这里，Windows系统调用指令指的是安卓系统调用指令转换后的Windows系统调用指令，Windows NT内核需要判断将Windows系统调用指令调用的Linux内核的进程类型，再决定采取什么策略执行系统调用。在本发明一些实施例中，转换后的Windows系统调用指令对应的进程类型可以包括普通进程和实时进程。那么，当Windows系统调用指令的进程类型为普通进程时，采用分时调度策略执行系统调用，实现与系统调用对应的安卓系统功能；而当Windows系统调用指令的进程类型为实时进程时，采用实时调度策略执行系统调用。

[0138] 详细的，为了协调多个进程的“同时”运行，给进程定义优先级。如果有多个进程同时处于可执行状态，那么先执行优先级高的进程。在本发明一些实施例中，针对于普通进程，使用SCHED_NORMAL(分时调度策略)调度策略执行系统调用，针对实时进程选SCHED_FIFO或SCHED_RR调度策略(实时调度策略)执行系统调用。实时进程的优先级都高于普通进程，在调度策略中，实时进程只会被高级的实时进程抢占，同级的实时进程直接是按照FIFO或者RR规则调度的。对于普通进程，SCHED_NORMAL调度策略提高交互式应用的优先级，使得进程能更快地被调度。

[0139] 需要说明的是，如果一个进程有实时需求(即是一个实时进程)，则只要是可执行状态的，以驱动的方式实现的AOW引擎就一直执行该实时进程，以尽可能地满足实时进程对CPU的需要，直到完成所需要做的事情，然后睡眠或退出(变为非可执行状态)。而如果有多个实时进程都处于可执行状态，则AOW引擎会先满足优先级最高的实时进程对CPU的需要，直到优先级最高的实时进程变为非可执行状态。

[0140] 可以理解的是，只要高优先级的实时进程一直处于可执行状态，低优先级的实时进程就一直不能得到CPU；只要一直有实时进程处于可执行状态，普通进程就一直不能得到CPU。

[0141] 如果多个相同优先级的实时进程都处于可执行状态，这时就有两种调度策略可供选择：

[0142] 1、SCHED_FIFO：先进先出。直到先被执行的进程变为非可执行状态，后来的进程才被调度执行。在这种策略下，先来的进程可以执行sched_yield系统调用，自愿放弃CPU，以让权给后来的进程；

[0143] 2、SCHED_RR：轮转调度。AOW为实时进程分配时间片，在时间片用完时，让下一个进程使用CPU。

[0144] 需要说明的是，这两种调度策略仅仅针对于相同优先级的多个实时进程同时处于可执行状态的情况。在安卓系统中，用户态的应用程序可以通过sched_setscheduler系统调用来设置进程的调度策略以及相关调度参数；sched_setparam系统调用则只用于设置调度参数。这两个系统调用要求进程具有设置进程优先级的能力(CAP_SYS_NICE，一般来说需要root权限)。通过将进程的策略设为SCHED_FIFO或SCHED_RR，使得进程变为实时进程，而

进程的优先级则是通过以上两个系统调用在设置调度参数时指定的。

[0145] 实时进程调度是让处于可执行状态的最高优先级的实时进程尽可能地占有CPU，因为它有实时需求;而普通进程则被认为是没有实时需求的进程,于是调度程序力图让各个处于可执行状态的普通进程通过CFS调度器实现和平共处地分享CPU(例如,SCHED_NORMAL),从而让用户觉得这些进程是同时运行的。

[0146] 可以理解的是,本发明实施例从软件的角度,对Windows系统、Linux内核和安卓系统深入研究,打造出Windows和安卓之间的一层介质,使得LinuxELF二进制文件可以运行在Windows系统上,AOW引擎通过Windows NT内核实现出Linux内核来执行Linux ELF二进制文件,即AOW引擎利用Windows NT内核提供的API以及编程功能,实现一套Linux内核(即内核功能,是内核函数的封装)和Linux的系统调用(函数)。当系统调用请求是由Linux ELF二进制文件发出时,AOW引擎会实时地将Linux内核的系统调用(函数)转换为实现等效的Window NT的函数的调用,对于Windows NT内核中没有与系统调用函数等效的函数API的情况,则采用编程的方式预先实现Windows NT内核中等效的API,从而完成把安卓底层的系统调用翻译(转换)成Windows API的工作。本方案没有用到虚拟化(机)技术,所以会非常的节省资源且流畅地运行安卓应用,包括游戏和软件。

[0147] 本发明实施例中,不仅要考虑做成实现引擎后带给用户的更愉悦的游戏体验,更要综合考虑Windows系统、安卓系统和Linux内核的不同,尤其在开发中Linux内核的进程调度、内存管理、文件系统、网络和渲染等系统架构模块以及系统调用与Windows系统的差异性。本发明实施例中的实现引擎,在给用户增加新的功能体验的同时,不降低高品质的游戏畅玩效果。

[0148] 作为示例,在PC Windows实现安卓系统,使得用户可以在PC电脑上畅玩安卓游戏,利用电脑的硬件配置运行手机游戏,同时配合键盘和鼠标的操作,大大的提高了游戏体验,同时不会影响到游戏本身,如画质输出、流畅度等等。

[0149] 可以理解的是,在引擎中执行安卓系统指令到视窗系统指令的转换,并通过执行转换得到的视窗系统指令的方式运行安卓系统,从视窗系统的指令执行的层面实现安卓系统,这种底层的实现方式相较于相关技术在视窗系统中虚拟安卓系统效率更高,还降低了硬件资源的要求;在引擎中利用自实现的系统调用和安卓系统内核来完成安卓应用的系统调用请求,由于不需要依赖于第三方库,相较于相关技术提供的虚拟化方案,克服了对第三方库的依赖,因而能够在各种版本的视窗系统中运行安卓应用,具有良好的兼容性。

[0150] 基于上述实现的基础,本发明实施例提供的跨系统运行安卓应用的方法还可以包括:步骤S107-步骤S108,以及步骤S109。如下:

[0151] 步骤S107、在引擎的窗体进程中显示的安卓应用的界面,接收对应安卓应用的界面的操作。

[0152] 步骤S108、当响应操作需要使用安卓系统内核功能时,通过引擎中实现的安卓系统内核的系统调用接口,发起与安卓系统内核功能对应的系统调用请求。

[0153] AOW引擎在安卓系统启动后,就可以在AOW引擎的窗体进程中显示安卓应用的界面,当用户想要在安卓系统中实现安卓应用的功能时,会通过窗体进程中显示的安卓应用的界面,接收对应安卓应用的界面的操作。于是,引擎响应于完成操作执行的指令,当响应操作需要使用安卓系统内核功能时,通过引擎中实现的安卓系统内核的系统调用接口,开

始发起与安卓系统内核功能对应的系统调用请求，这样该AOW引擎响应于该系统调用请求，利用在引擎中实现的系统调用函数，以及实现的安卓系统内核功能完成系统调用请求，得到系统调用结果，最后向安卓应用返回系统调用结果，完成一次安卓系统的系统调用。

[0154] 需要指出，一个安卓应用的功能的实现，可能需要多个系统调用和安卓内核功能一起配合实现，每次系统调用的流程是一致的，因此根据本发明实施例提供了详细的一次系统调用的实现流程，可以轻易实施一系列的系统调用。

[0155] 步骤S109、当系统调用结果表示系统调用请求成功时，在窗体进程中显示系统调用请求的响应界面。

[0156] AOW引擎向安卓应用返回系统调用结果，当系统调用结果表示系统调用请求成功时，表征通过该系统调用配合实现的安卓应用的功能是可以成功实现的，于是，该AOW引擎可以在窗体进程中显示系统调用请求的响应界面。

[0157] 基于上述实施例的实现，在本发明一些实施例中，当使用本发明实施例提供的安装有AOW引擎的终端时，在终端启动或开机时，接收启动跨系统运行安卓应用的引擎的指令，根据指令在视窗系统中执行引擎的启动操作；在AOW引擎启动后的进程中，转换安卓系统指令为等效的视窗系统指令，并通过执行视窗系统指令的方式运行安卓系统。在启动了安卓系统后，在窗体进程中显示的安卓应用界面上对安卓应用进行使用，接收操作，即在安卓应用界面中接收操作，响应于完成操作执行的指令需要使用安卓系统内核功能，通过安卓系统内核的系统调用接口，发起与安卓系统内核功能对应的系统调用请求，当解析到发起系统调用请求的指令时，执行针对引擎中实现的系统调用函数、以及实现的安卓系统内核功能的调用操作；其中，调用操作用于完成安卓应用通过系统调用请求所请求实现的功能，得到系统调用结果；向安卓应用返回系统调用结果，当系统调用结果表示在系统调用请求成功时，在窗体进程中显示系统调用请求的响应界面，该响应界面为指令触发的功能对应的当前应用界面。

[0158] 作为示例，假设安卓应用为网络游戏，当使用本发明实施例提供的安装有AOW引擎的终端时，在终端启动或开机时，接收启动跨系统运行安卓应用的引擎的指令，根据指令在视窗系统中启动跨系统运行安卓系统的引擎和安卓系统(引擎)。在启动了终端中的安卓系统后，在窗体进程中显示的网络游戏界面上的游戏登录界面，用户在游戏登录界面输入用户名和密码，点击确定登录按键生成登录指令，依据该登录指令对安卓系统发起登录系统调用等一系列请求，其中，登录系统调用需要通过安卓的内核功能网络与游戏服务器进行信息交互实现，因此，转换安卓系统调用为视窗系统的调用，实现通过视窗系统的网络功能(即网络通信)，配合登录指令需要的其他系统调用，从而完成通过与服务器的信息交互实现的登录过程。

[0159] 那么，上述示例性的举例的实现可以基于图10的硬件实体系统架构实现，在本发明一些实施例中，引擎可以安装在具有视窗系统的设备或终端中，这样的设备或终端中可以安装有安卓应用，那么，本发明实施例中安卓应用的一些功能实现还可以与服务器进行信息交互来完成。

[0160] 图10为本发明实施例中一种服务器与终端的信息交互的硬件实体系统架构图，可以基于如图10所示的具有Windows系统的终端2(例如电脑)与服务器3的架构，实现本发明实施例提供的跨系统运行安卓应用的方法。

[0161] 其中，服务器3为应用服务器，该服务器3与终端2通过网络4实现应用功能，网络3中包括路由器，网关等等网络实物，图中并未体现。终端的类型，可以包括具有视窗系统的手机、平板电脑或PDA、台式机、PC机、智能TV等类型，即在本发明一些实施例中的终端可以为具有Windows系统可安装安卓应用的终端设备，本发明实施例不作限制。

[0162] 其中，终端中可以安装有各种用户所需的安卓应用，比如具备娱乐功能的应用(如视频应用，音频播放应用，游戏应用和阅读软件)，又如具备服务功能的应用(如地图导航应用、团购应用和拍摄应用等)，本发明实施例不作限制。

[0163] 作为示例，如图11所示，在Windows系统的运行界面1中，启动AOW引擎后，在AOW引擎对应的窗体进程中显示安卓系统的运行界面2(也可以为安卓应用的运行界面)，当想要在安卓系统的运行界面2中实现文件管理功能时，可以通过操作该安卓系统的运行界面2上的文件管理图标实现，结合图9，就是通过执行Is命令，去AOW引擎中的Linux内核中实现文件管理的功能的系统调用，并将系统调用结果返回给用户态，即在安卓系统的运行界面2的区域显示文件目录3(例如，图片、音频、视频、文档、压缩包和应用等等)，以便用户在文件目录3中选择查找所需的文件。

[0164] 可以理解的是，在引擎中执行安卓系统指令到视窗系统指令的转换，并通过执行转换得到的视窗系统指令的方式运行安卓系统，从视窗系统的指令执行的层面实现安卓系统，这种底层的实现方式相较于相关技术在视窗系统中虚拟安卓系统效率更高，还降低了硬件资源的要求；在引擎中利用自实现的系统调用和安卓系统内核来完成安卓应用的系统调用请求，由于不需要依赖于第三方库，相较于相关技术提供的虚拟化方案，克服了对第三方库的依赖，因而能够在各种版本的视窗系统中运行安卓应用，具有良好的兼容性。

[0165] 本发明实施例提供了一种计算机可读存储介质，存储有可执行指令，用于引起一个或多个处理器执行前述实施例中的跨系统运行安卓应用的方法。

[0166] 其中，计算机可读存储介质位于存储器中，存储器可以是磁性随机存取存储器(ferromagnetic random access memory，FRAM)、只读存储器(Read Only Memory，ROM)、可编程只读存储器(Programmable Read-Only Memory，PROM)、可擦除可编程只读存储器(Erasable Programmable Read-Only Memory，EPROM)、电可擦除可编程只读存储器(Electrically Erasable Programmable Read-Only Memory，EEPROM)、快闪存储器(Flash Memory)、磁表面存储器、光盘、或只读光盘(Compact Disc Read-Only Memory，CD-ROM)等存储器。

[0167] 本领域内的技术人员应明白，本发明的实施例可提供为方法、系统、或计算机程序产品。因此，本发明可采用硬件实施例、软件实施例、或结合软件和硬件方面的实施例的形式。而且，本发明可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器和光学存储器等)上实施的计算机程序产品的形式。

[0168] 以上所述，仅为本发明的较佳实施例而已，并非用于限定本发明的保护范围。

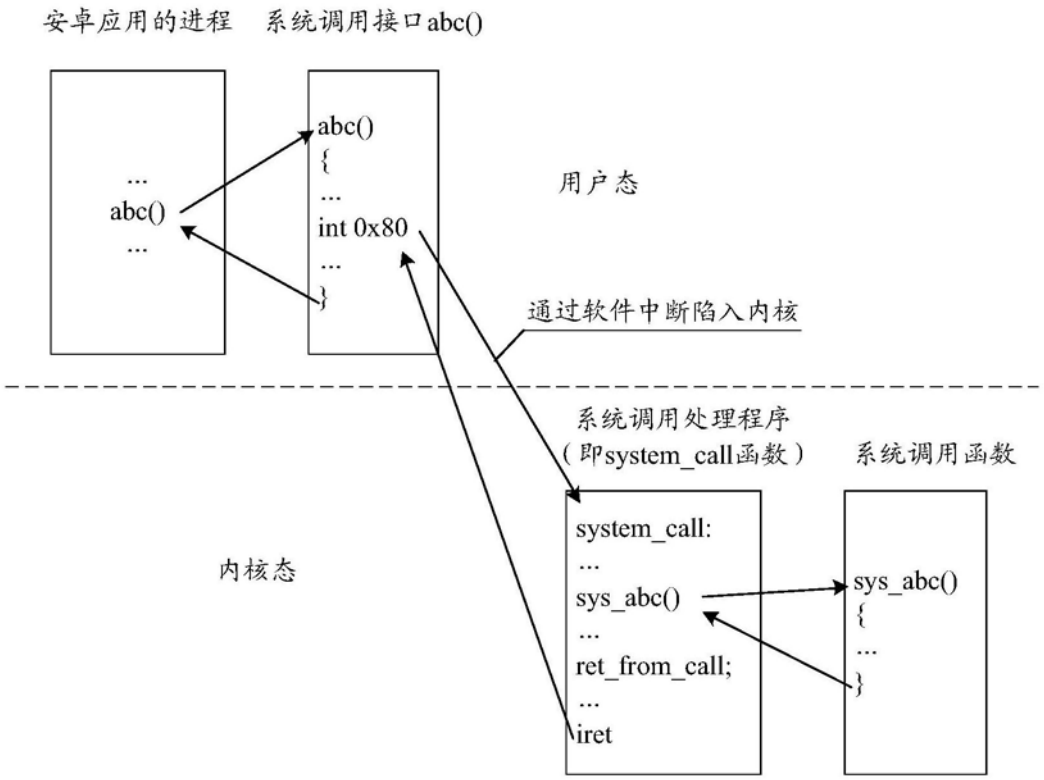

安卓应用的进程
系统调用接口abc()
...
abc()
...
abc()
{
...
int 0x80
...
}
用户态
通过软件中断陷入内核
系统调用处理程序
(即system_call函数)
系统调用函数
内核态
system_call:
...
sys_abc()
...
ret_from_call;
...
iret
sys_abc()
{
...
}

图1

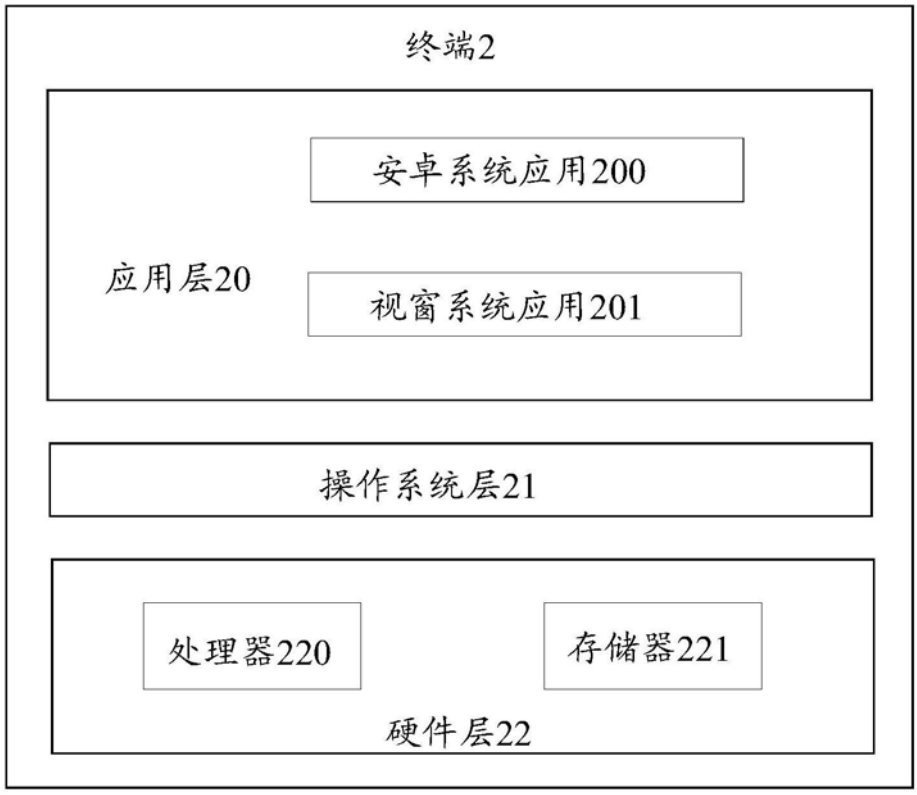

终端2
安卓系统应用200
应用层20
视窗系统应用201
操作系统层21
处理器220
存储器221
硬件层22

图2-A

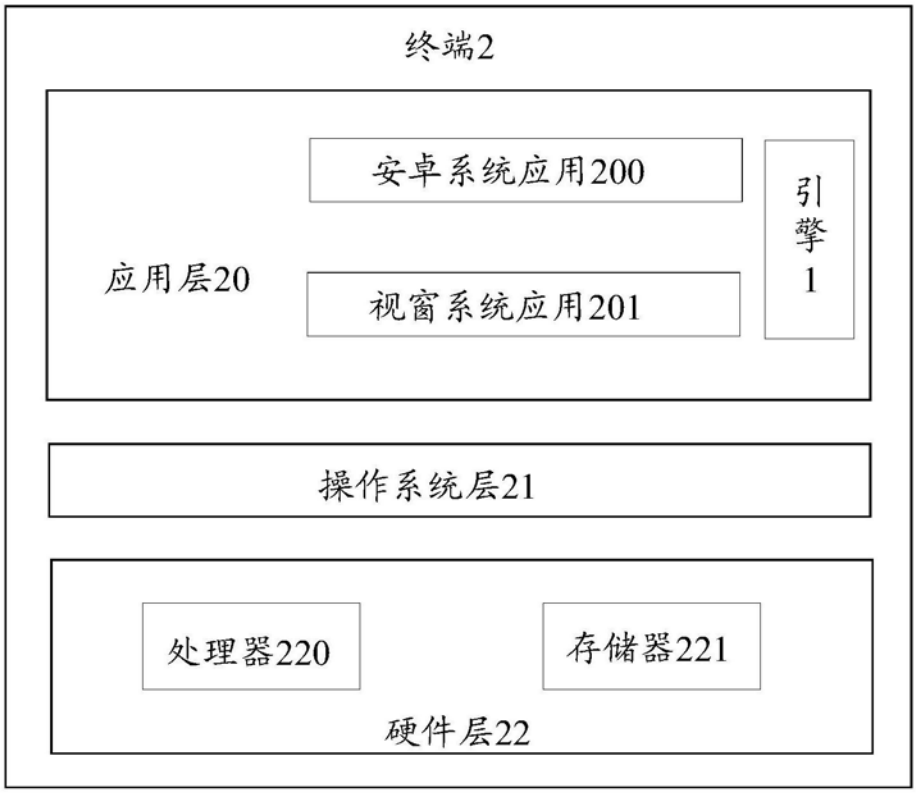

终端2
安卓系统应用200
引擎1
应用层20
视窗系统应用201
操作系统层21
处理器220
存储器221
硬件层22

图2-B

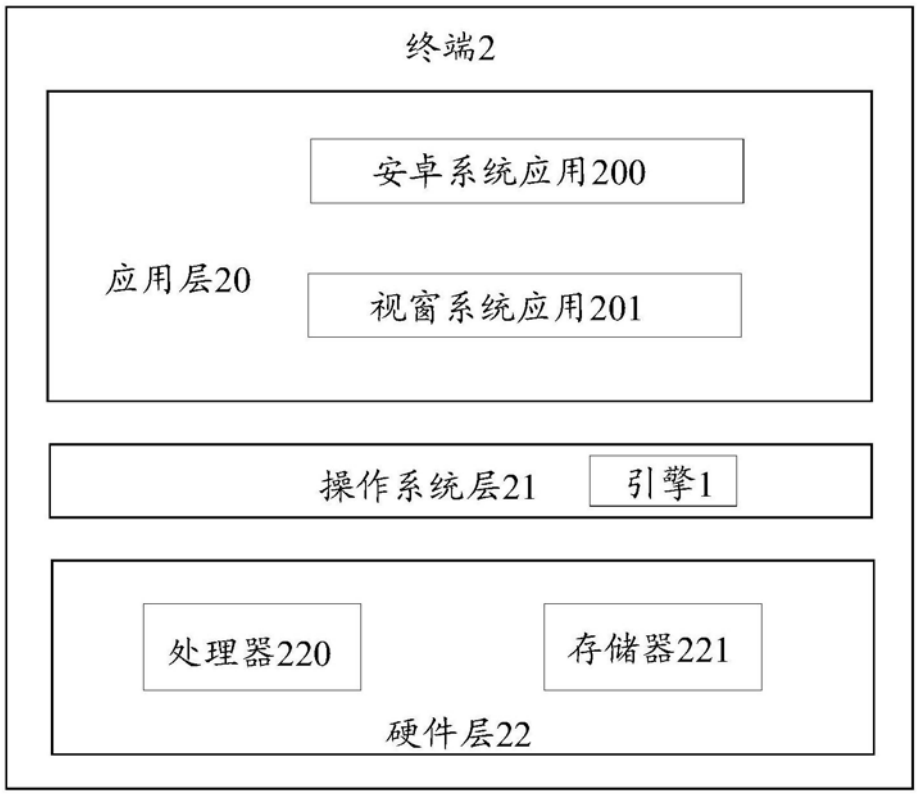

终端2
安卓系统应用200
应用层20
视窗系统应用201
操作系统层21
引擎1
处理器220
存储器221
硬件层22

图2-C

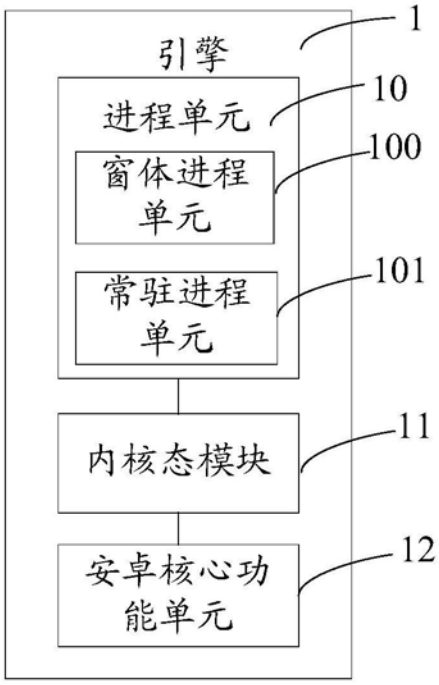



图3

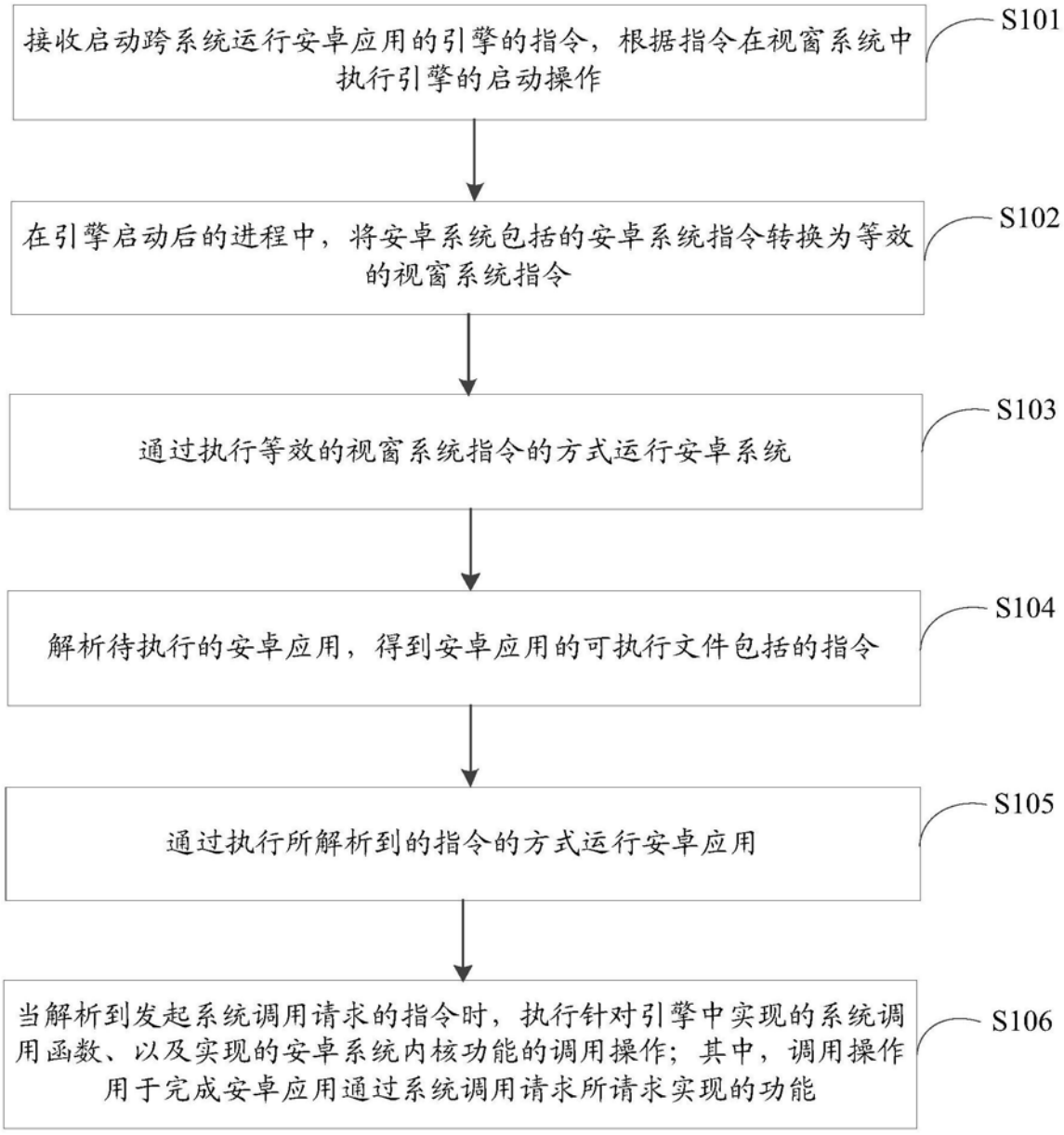

接收启动跨系统运行安卓应用的引擎的指令，根据指令在视窗系统中执行引擎的启动操作
S101
在引擎启动后的进程中，将安卓系统包括的安卓系统指令转换为等效的视窗系统指令
S102
通过执行等效的视窗系统指令的方式运行安卓系统
S103
解析待执行的安卓应用，得到安卓应用的可执行文件包括的指令
S104
通过执行所解析到的指令的方式运行安卓应用
S105
当解析到发起系统调用请求的指令时，执行针对引擎中实现的系统调用函数、以及实现的安卓系统内核功能的调用操作；其中，调用操作用于完成安卓应用通过系统调用请求所请求实现的功能
S106

图4

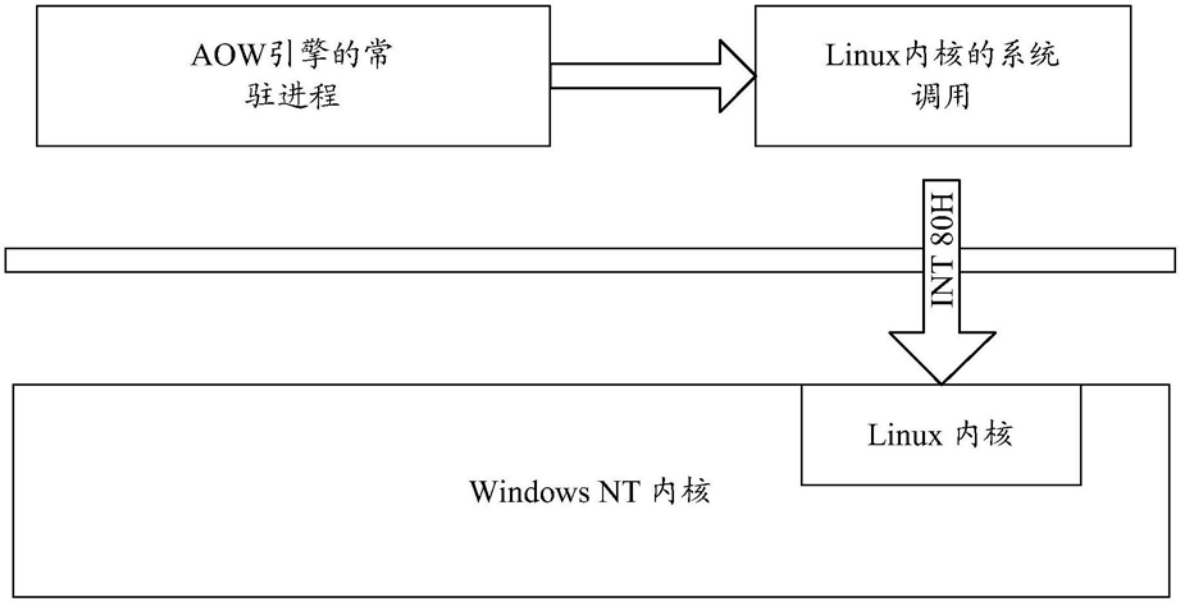

AOW引擎的常
驻进程
Linux内核的系统
调用
INT 80H
Linux 内核
Windows NT 内核

图5

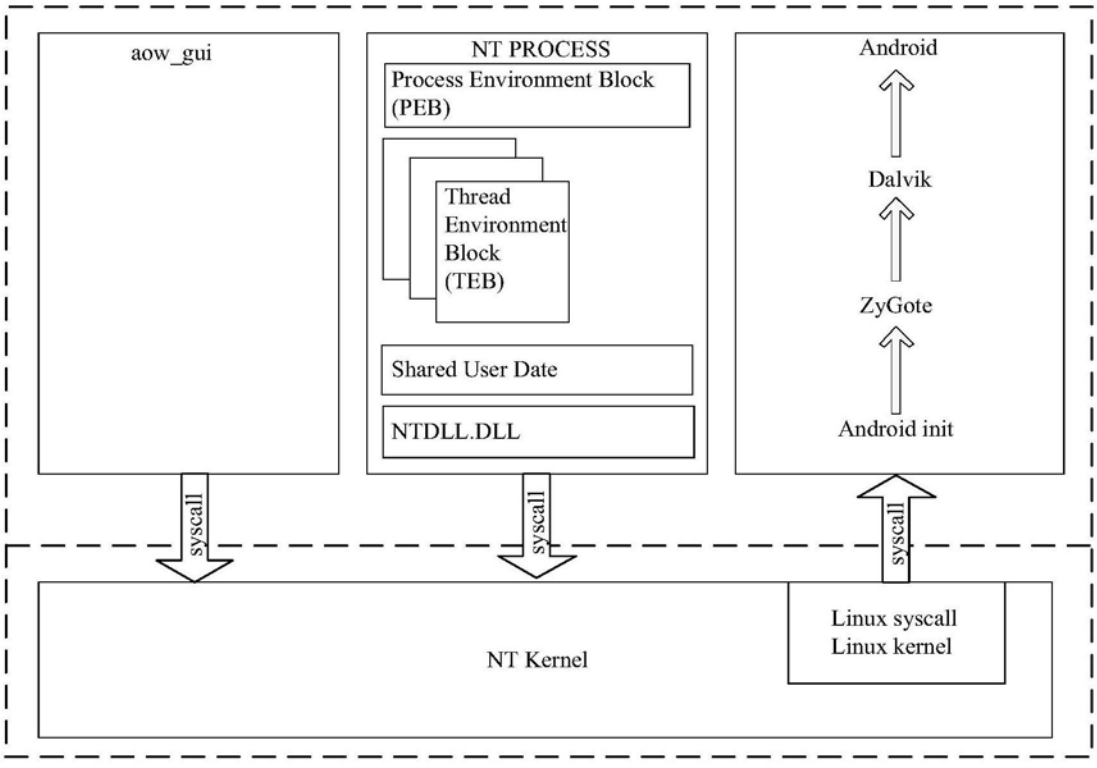

aow_gui
NT PROCESS
Process Environment Block
(PEB)
Thread
Environment
Block
(TEB)
Shared User Date
NTDLL.DLL
Android
Dalvik
ZyGote
Android init
syscall
syscall
syscall
NT Kernel
Linux syscall
Linux kernel

图6

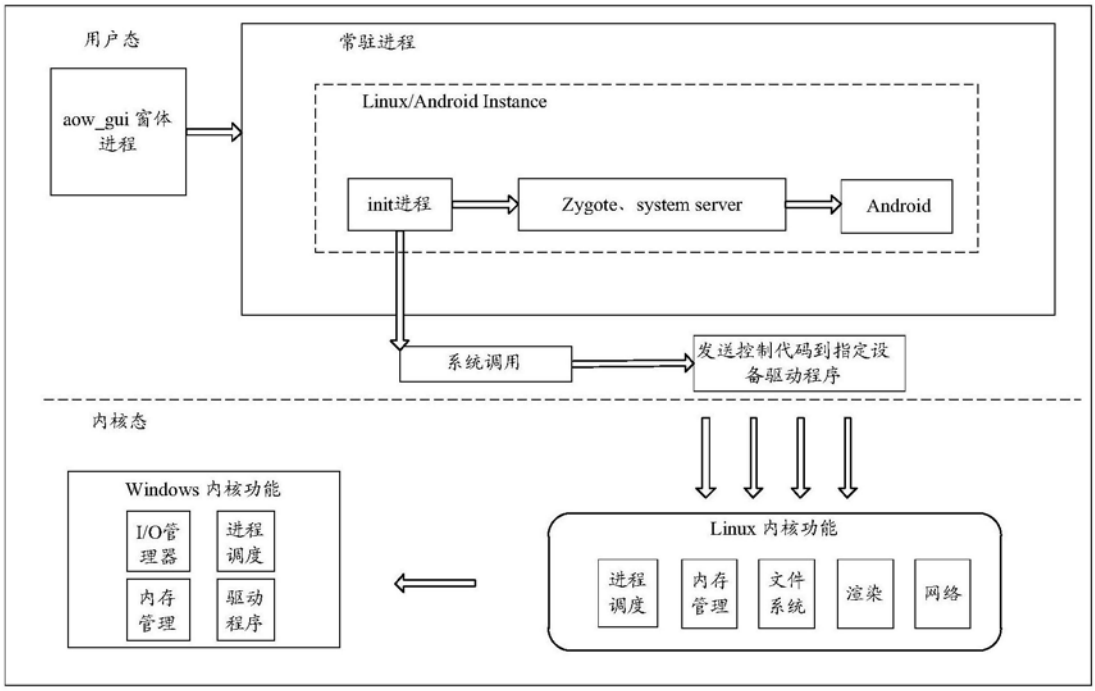

用户态
aow_gui 窗体进程
常驻进程
Linux/Android Instance
init进程
Zygote、system server
Android
系统调用
发送控制代码到指定设备驱动程序
内核态
Windows 内核功能
I/O管理器
进程调度
内存管理
驱动程序
Linux 内核功能
进程调度
内存管理
文件系统
渲染
网络

图7

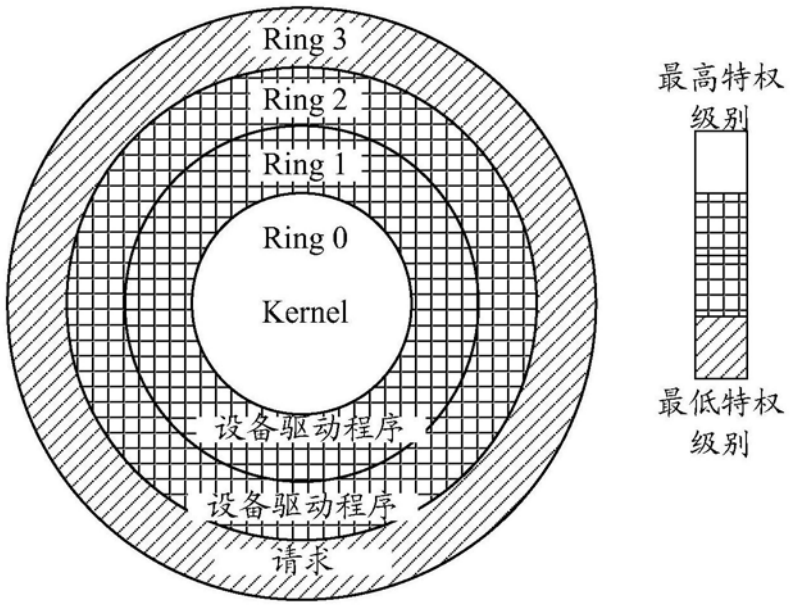

Ring 3
Ring 2
Ring 1
Ring 0
Kernel
设备驱动程序
设备驱动程序
请求
最高特权级别
最低特权级别

图8

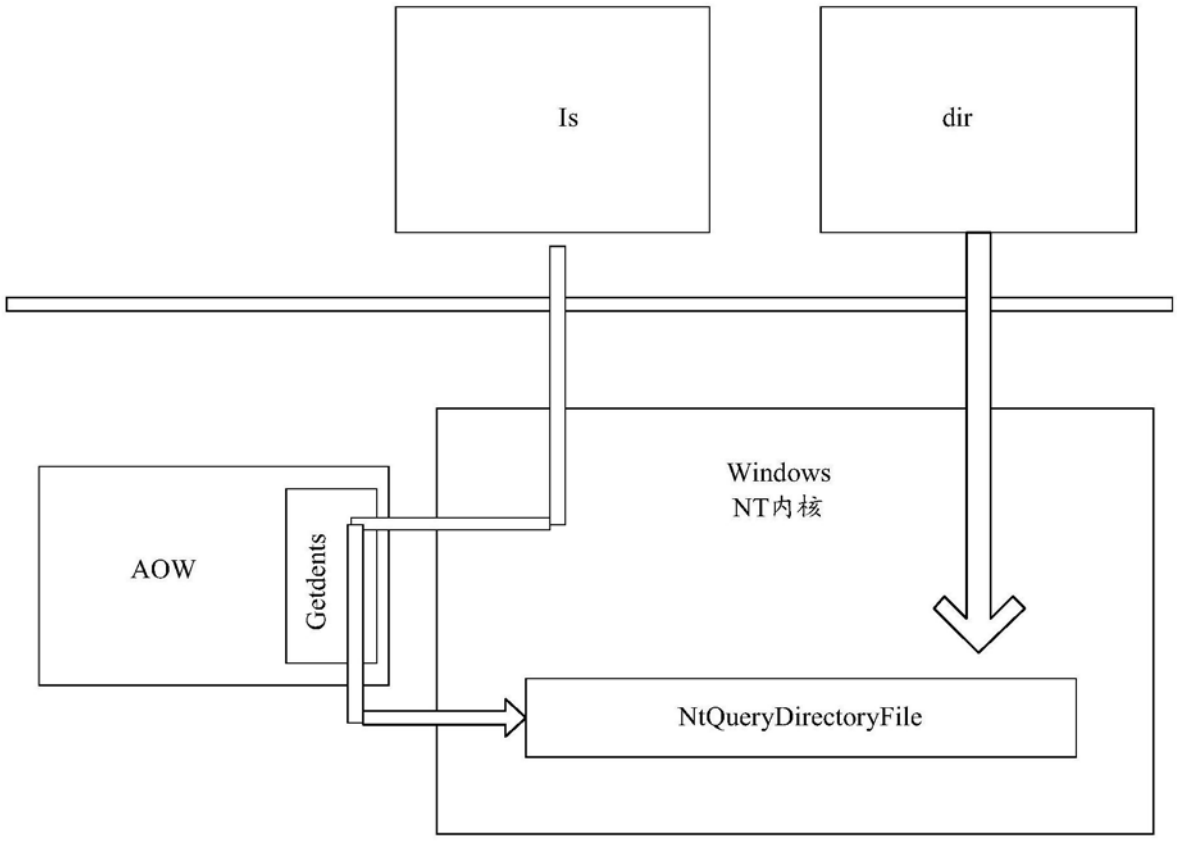



图9

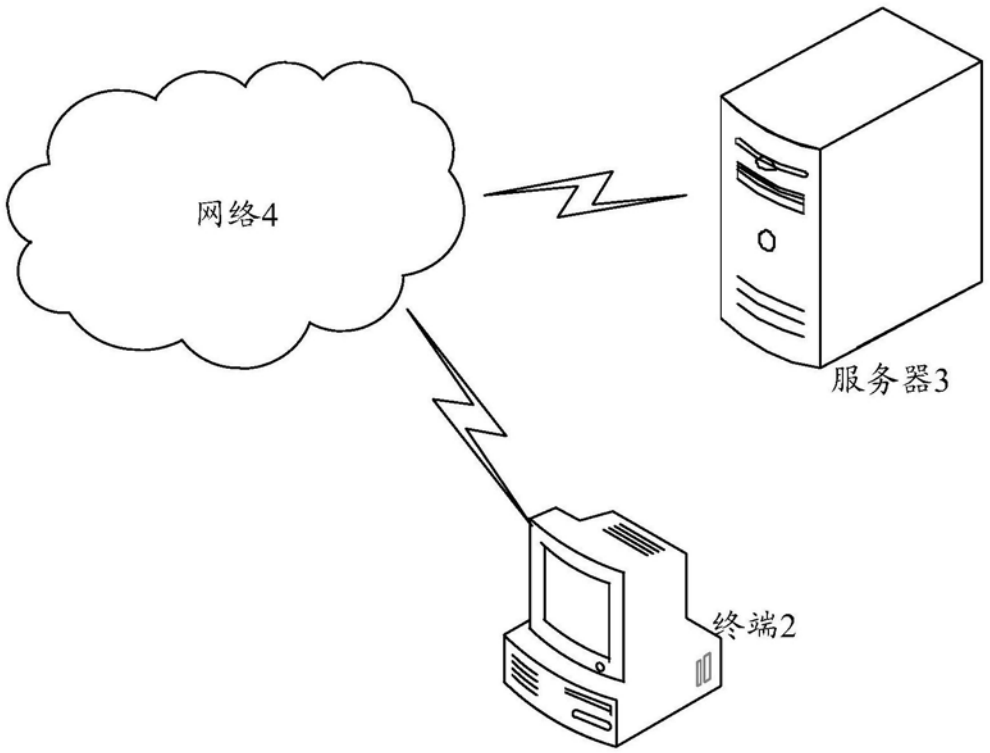



图10

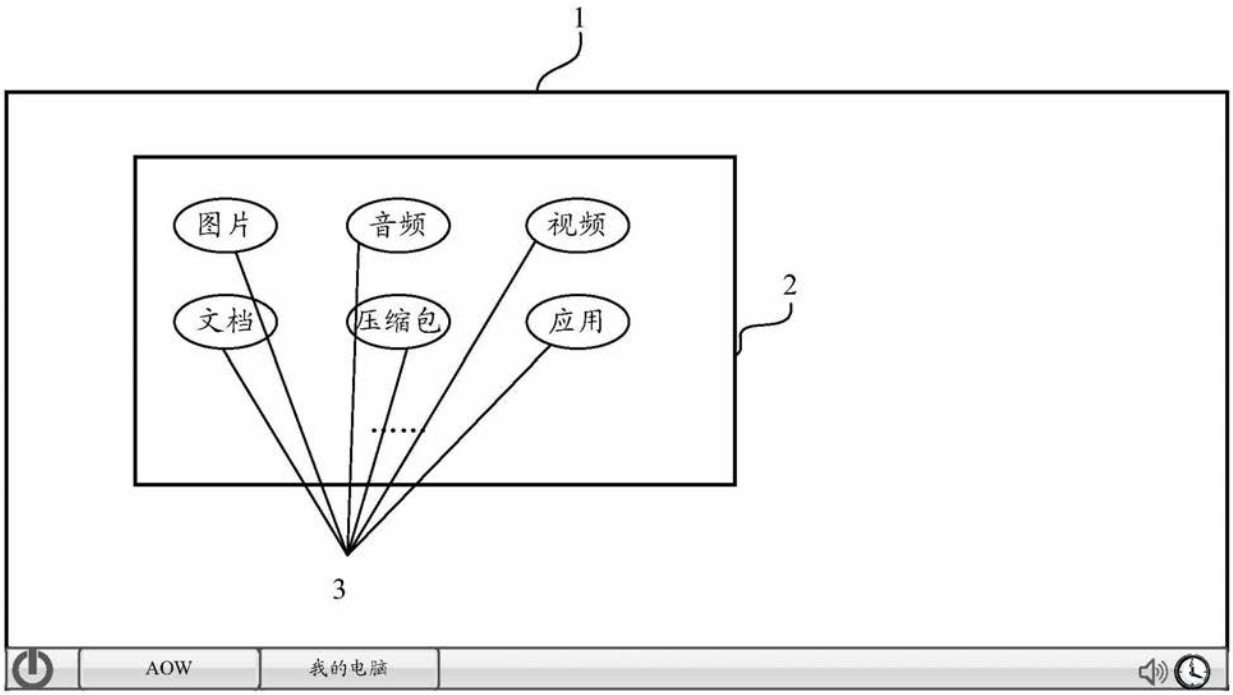



图11